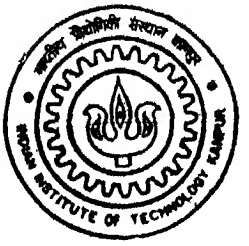
A Virtual Instrumentation Implementation for remote monitoring and analysis of Electric Power Substation parameters

by
Narayana Murthy C B N S

TH
EE/2000/M
M969v



DEPARTMENT OF ELECTRICAL ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY KANPUR
February 2000

A Virtual Instrumentation Implementation for remote monitoring and analysis of Electric Power Substation parameters

A Thesis Submitted

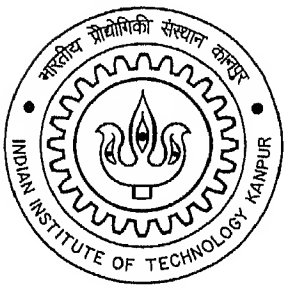
In Partial Fulfillment of the Requirements

For the Degree of

MASTER OF TECHNOLOGY

By

Narayana Murthy C B N S



to the

DEPARTMENT OF ELECTRICAL ENGINEERING

INDIAN INSTITUTE OF TECHNOLOGY KANPUR

February 2000

11 MAY 2000 EE
CENTRAL LIBRARY
IIT KANPUR
A 130799

TH
EE/2000/M
M969v

A130799

CERTIFICATE

This is to certify that the work contained in this thesis entitled *A Virtual Instrumentation Implementation for remote monitoring and analysis of Electric Power Station parameters,* by Narayana Murthy C B N S has been carried out under my supervision and that this work has not been submitted elsewhere for a degree

K R Srivathsan 27-2-2000

Dr K R Srivathsan
Department of Electrical Engineering
Indian Institute of Technology Kanpur

Acknowledgement

I take this opportunity to thank Dr K R Sreenathan for his guidance and encouragement during the course of this thesis work I would also like to thank all my teachers at IIT Kanpur

I would like to thank Mr Reddy Mr Amitabh Roy Mrs Neeru Chabra for the suggestions in the thesis work I would like to thank Dr Sanjay Gupta for clarifying my technical doubts in LabVIEW Also I would like to thank Mr Vivek Mudgil Vineet Akanksha for their help in the Lab

Above all I would like to thank my friends Murali Shankar Divyesh Swaroop Sudheer Ranga Boddu Praveen and many more who made my stay wonderful at hall5

Finally its time to thank my classmates Gaurav Praveen Rao Harish Dhaval Alpna Gupta Bunnu and Dubey

Contents

List of Figures

Abstract

Geographically distributed networks such as power distribution water gas etc have not seen the widespread use of networked installation for monitoring and control IIT Kanpur has developed a sophisticated monitoring and control network with associated devices for the electric power distribution sector Monitoring and control is done from a communication controller The communication controller can be connected over the Internet This gives an opportunity for remote visualization of the power distribution utility parameters

The existing interface for monitoring and control and its communication has been developed using object oriented approaches It addresses objects in a format that is not compatible with open Internet standards This interface also lacks live instrument display of utility parameters The availability of virtual instrumentation based packages such as LabVIEW enables live instrument display of important utility parameters The LabVIEW based interface can communicate with remote systems over the Internet using its TCP/IP features Further LabVIEW can be used as an online diagnostic tool with appropriate remote Data Acquisition system

The necessary data structures for the substation elements are assumed to be available in ASN 1 format In practice this assumption amounts to developing a translator for the data collected in proprietary/non open format to that of ASN 1 In this thesis an implementation of converting c data structures into ASN 1 has been done with a Snacc compiler The front panels of the substation elements are made with the G language code The user interface is developed in a manner that if a user selects an option at the layout diagram the corresponding panel pops up on the screen A final implementation is done using simulated data for the remote visualization of the electric substation parameters

Chapter 1

Introduction

Major process industries are usually equipped with proprietary networks for monitoring and control supplied by industrial automation equipment manufacturers In recent times attempts have been made to provide open standards such as Field Bus [1] for such applications Geographically distributed individual utilities such as power distribution gas water etc have not seen widespread use of networked installation for monitoring and control With the cost of network components and embedded systems falling down and wider choice of multi access communication technologies such as wireless and many wired networks becoming available it has become attractive to utilize them for distributed monitoring and control of utilities

With the above in view IIT Kanpur has developed a sophisticated monitoring and control network with associated devices for the electric power distribution sector referred to hereafter as the Power Distribution Automation System (PDAS) The existing GUI and communication for this application has been developed using conventional OOP approaches over a standalone system communicating with the remote terminal units (RTUs) through a communication controller (CC) The CC can be connected to the Internet With such an Internet linked PDAS two important issues arise First is to provide a web based platform for remote monitoring and visualization of the PDAS This work is being studied as a separate thesis The second is the need for remote instrumentation like visualization of the parameters (such as voltages currents power phase angle etc) associated with subsystems (such as transformers feeders switches etc) that are connected to the PDAS Both the issues referred above have some commonality of using the underlying data acquired from the PDAS There is considerable value in integrating open Internet compatible GUI using web based user interface or popular

packages such as the LabVIEW [2] based virtual instrumentation The latter with its rich set of processing analysis and live instrument display of important utility parameters offers rich scope for being used as on line diagnostic distributed instrumentation In this thesis an attempt has been made to configure and tailor a LabVIEW centered instrumentation for monitoring and analysis of distributed real time utility parameters over TCP/IP based networks It is assumed that the actual data collection is done by a communication and control system that acquires data in some industrial standard compatible format over a network of Remote Terminal Units (RTU) The monitoring network may or may not be based on TCP/IP

1 1 Monitoring of Industrial utilities

The figure shown in the next page describes the approach used for monitoring geographically distributed industrial utilities The access network within the industry may be a wired or wireless network The wired networks such as Ethernet and Fast Bus are commonly used in the industries The access network is the network of Remote terminal units (RTUs) The RTU is the interface between the access network and the process It can communicate in a bi directional manner enabling the process to be either controlled or monitored The communication controller (CC) is an entity which interacts with the RTUs The communication controller polls the data from the RTUs Typically a user interface is located at the communication controller enabling monitoring and control over the utility network A typical communication controller can span a small geographical area and many such communication controllers exist in the case of power distribution automation

With the Internet technology it is possible to connect the communication controllers through the Internet Monitoring can be achieved from web based or LabVIEW based monitor through communication between the CCs via the Internet

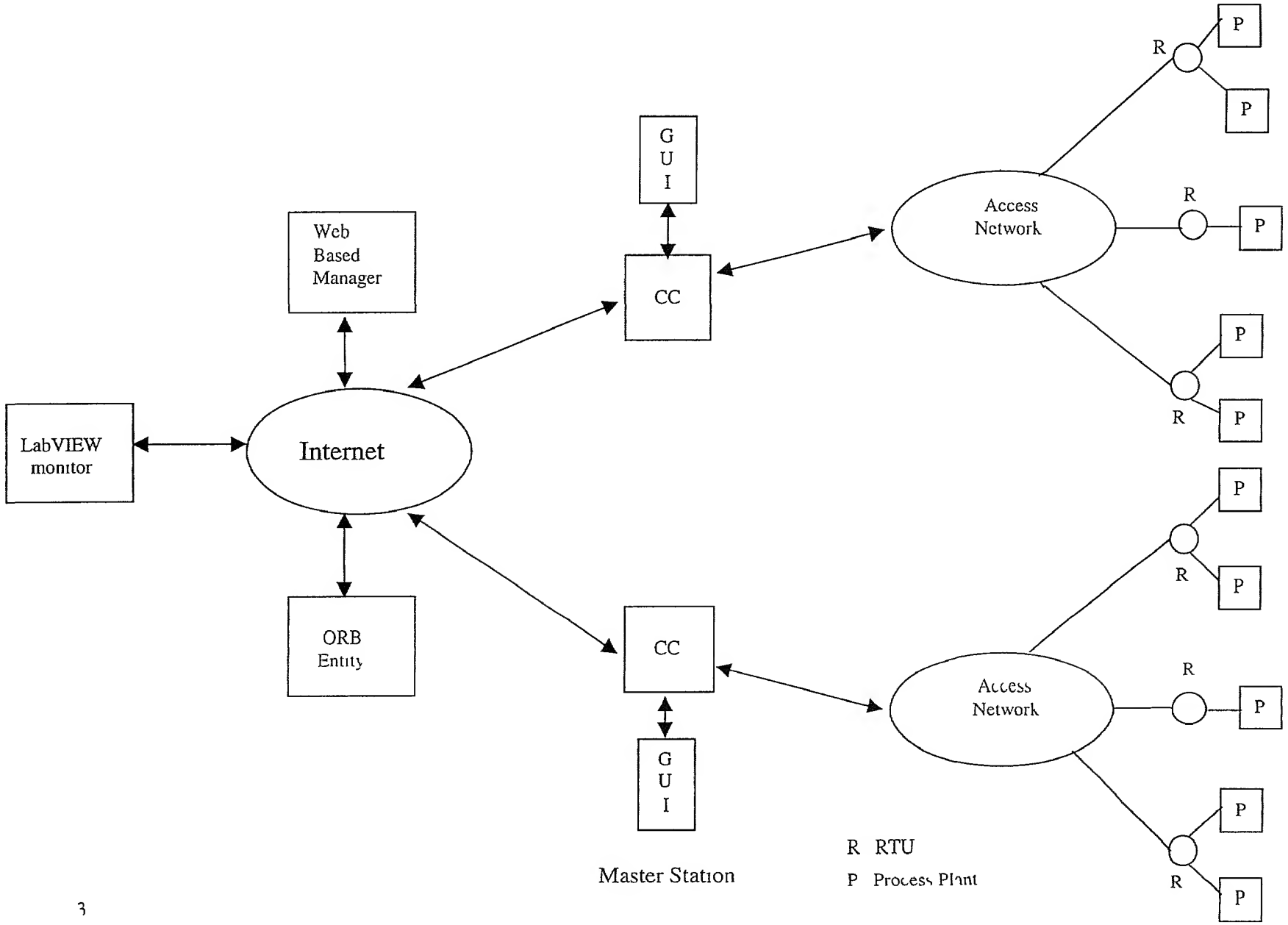


Figure 1 1 Distributed network architecture for remote monitoring and control

1 2 Issues in Real Time Monitoring over TCP/IP

Real time monitoring and control can be possible if a correct picture of the field is available This is possible only if the number of samples taken are large and the sampled data can be processed within negligible time This requires not only fast processing equipment for signals but also low communication delays Industrial standards like Field Bus satisfy the requirements and are in current use for real time monitoring and control

In the case of monitoring geographically distributed networks over TCP/IP Internet based networks the correct picture of the field parameters cannot be obtained This is due to reasonable communication delays caused by network work congestion As the communication delays cannot be neglected these geographical interconnections can be used for global observation and monitoring but without any control in strict real time

It is assumed that LabVIEW centered distributed monitoring and analysis of utilities will access real time data collected over the utilitys network over a TCP/IP based Internet type network There have been developments of using the Internet for distributed monitoring in recent times [3] In our scenario actual real time data is acquired at utilitys network level It is preprocessed partially there The LabVIEW display should provide reasonable real time presentation of important performance parameters and post analysis capability with acceptable delay that does not lead to a loss of time sense at the human operator level

1 3 LabVIEW and its TCP/IP capability

LabVIEW (Laboratory Virtual Instrument Engineering WorkBench) is a program development application much like the various commercial C or Basic development systems plus a GUI development system LabVIEW is different from these applications Other programming systems use text based languages to create

lines of code while LabVIEW uses a graphical programming language G to create programs in block diagram form LabVIEW programs are known as VIs LabVIEW uses terminology icons and ideas familiar to scientists and engineers and relies on graphical symbols rather than textual languages to describe programming actions LabVIEW has extensive libraries of functions and subroutines for most programming tasks LabVIEW contains application specific libraries for data acquisition GPIB and serial instrument control data analysis data presentation and data storage

LabVIEW has also libraries for networking and inter application communications [4] Several protocols are built into LabVIEW such as Transmission Control Protocol (TCP) User Data Protocol (UDP) Dynamic Data Exchange (DDE) Object Linking and Embedding (OLE) etc The communication features of LabVIEW are explained in chapter 3

For the remote monitoring of the power distribution automation system LabVIEW can be used as a diagnostic tool With the vast analysis vis available in LabVIEW one can measure the correlation between two waveforms find the ripple content of a particular bus voltage analyze the spectrum of a waveform have different types of graphs various kinds of indicating meters etc Therefore LabVIEW can act as an important tool for post analysis

1 4 Organization of the thesis

Chapter 2 describes the various frame works that can be used for geographically monitoring of industrial utilities Chapter 3 gives an introduction to virtual instrumentation and introduces the LabVIEW communication features Chapter 4 discusses the implementation of the thesis Chapter 5 summarizes the work and discusses the future aspects

Chapter 2

Open Frameworks for Remote monitoring

Traditionally distributed measurement and control operation in industries were supported by proprietary solutions from specialized vendors such as Honeywell Harris etc Minimal interface standards for component systems used in distributed industrial monitoring and control where available These were restricted to the lower layers physical and data link level only Field Bus [1] standards provided early attempts in bringing open architecture upto the network layer and supports for real time distributed tasks

With the rapid progress in Internet and platform independent approach to Internet Services Structure there arises a strong need to provide an open data base framework to address proprietary and open objects in the monitoring and control components of a distributed plant

In the recent years open framework for description of objects needed for the addressing of any objects constituted by collection of components has been made possible through standards such as X 700 [5] CORBA [6] and ASN 1 [7][8] Industries that develop and use the distributed measurement and control systems are migrating away from proprietary hardware and software platforms in favor of open systems and standardized approaches The open framework is necessary to ensure interoperability X 700 CORBA ASN 1 are necessary for open framework approaches in network management

ASN 1 is based on Abstract syntax approach and CORBA on the Interface Definition Language (IDL) CORBA is a complete architecture and

message passing specification in which the IDL and corresponding encodings form a relatively small part The CORBA IDL is simpler and less powerful than the ASN 1 notation and as a result encodings are generally much more verbose than the encodings of ASN 1 ASN 1 is generally used in protocol specifications where very general and flexible exchange of messages is needed between communicating partners whereas CORBA encourages a much more stylized invocation and response approach and generally needs a much more substantial supporting infrastructure

Network management as a term has many definitions depending on whose operational function is in question Network management in the power distribution automation consists of the status information and various other data of the power utilities

2 1 X 700 Object Model

2 1 1 Architecture

This architecture follows the description in [9]

OSI Network management is defined by the joint ISO/ITU T standard documents The approach taken by X 700 is to place an agent close to the information that is relevant to be managed or monitored

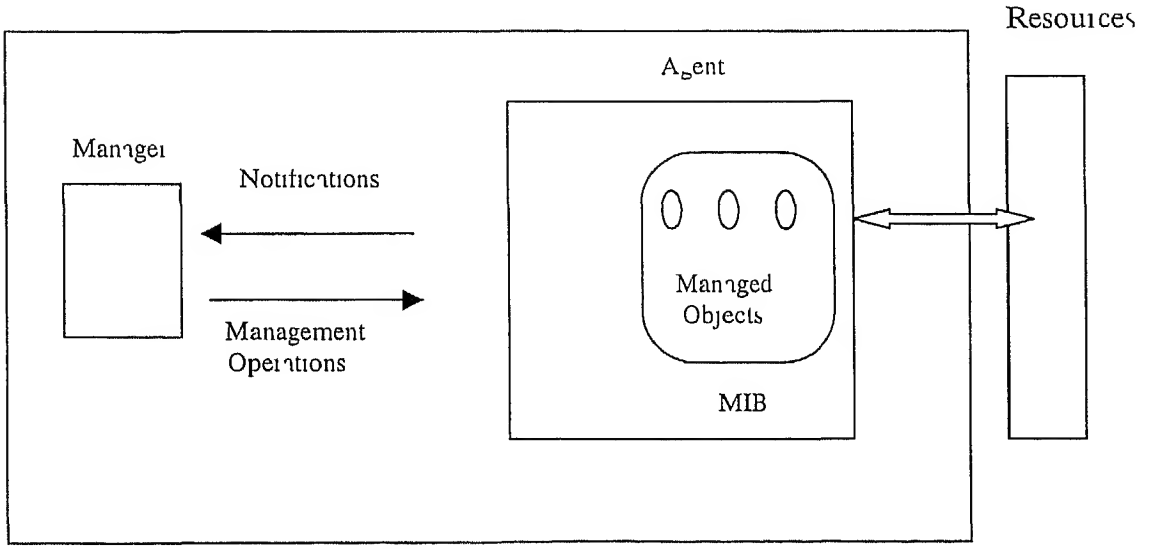


Figure 2 1 X 700 model

An agent maintains information about the state of the particular part of the network that it is responsible Queries about or changes of the state of the network can be sent to the agent and not directly to the objects The protocol used to interact with the agent is the Common Management Information Protocol (CMIP) CMIP is designed to provide a mechanism between an entity in the manager role and a system which contains the managed object representation of resources and the facilities needed to support management CMIP works at an application level The manager can send messages and receive replies over this protocol to manage the agent An agent may also send unsolicited information asynchronously to the manager e g to notify it of a problem in case of an element failure The common management Information Service Entity (CMISE) specifies seven requests that define the interaction between the manager and the agent CMISE offers the set of services which map onto the interface requirement of managed objects The general format of an operation is

{operation object reference access right token parameter list}

The seven requests that define the interaction between the manager and the client are

- M CREATE used to add management information to be maintained by the agent
- M DELETE removes management information from the agent
- M SET changes information in an agent
- M GET retrieves information from an agent
- M CANCEL GET since the amount of management information returned by an agent in response to an M Get request may be too large the request can be cancelled using this command
- M ACTION when the semantics of M Get or M Set to retrieve or change information are not powerful enough it is possible to specify operations on managed objects which can be called using M Action

- M EVENT REPORT if an error occurs in the network being managed an agent may at any time send an unsolicited message about the cause of the error using the M Event Report request

2 1 2 Object Model

The object model is taken from [5] Information to be managed by an agent is modeled as managed objects A managed object may represent either a logical resource such as an identification number or a real resource such as a switch A resource may be mapped to several managed objects or several resources may be mapped to one single managed object

A managed object contains attributes actions and notifications The type of attributes contained within a managed object is defined using Abstract Syntax Notation One (ASN 1) ASN 1 defines atomic types such as integers real or strings and aggregate types such as lists structures and unions

2 2 The CORBA Object Model

2 2 1 Architecture

The architecture follows the description given in [6] Object management groups Common Object Request Broker Architecture specifies the architecture by which instances in a heterogeneous environment can communicate with each other regardless of whether they are local or remote

The Object Request Broker (ORB) is responsible for accepting requests from clients finding the target object and its implementations invoking the request on it and returning the request to the caller The interface the client sees is completely independent of where the object is located or what programming language it is implemented in The ORB maintains an Interface Repository which

is essentially meta information about the interfaces registered with the ORB and the Implementation Repository which is the information about where the implementations of the interfaces are located The Interface Repository is needed by the ORB to marshal and unmarshall requests to and from clients and the Implementation Repository is needed for locating the implementations inorder to invoke requests on instances Clients call methods of other instances using Interface Definition Language (IDL) compiler generated client stubs which have to be included at compile time or the Dynamic Invocation Interface which allows clients to create request to be sent to instances at run time The interfaces are specified using the Interface Definition Language used to fully define all the aspects of the interface but no aspects of the implementation Implementation skeletons are IDL compiled generated implementations of the services offered by an interface

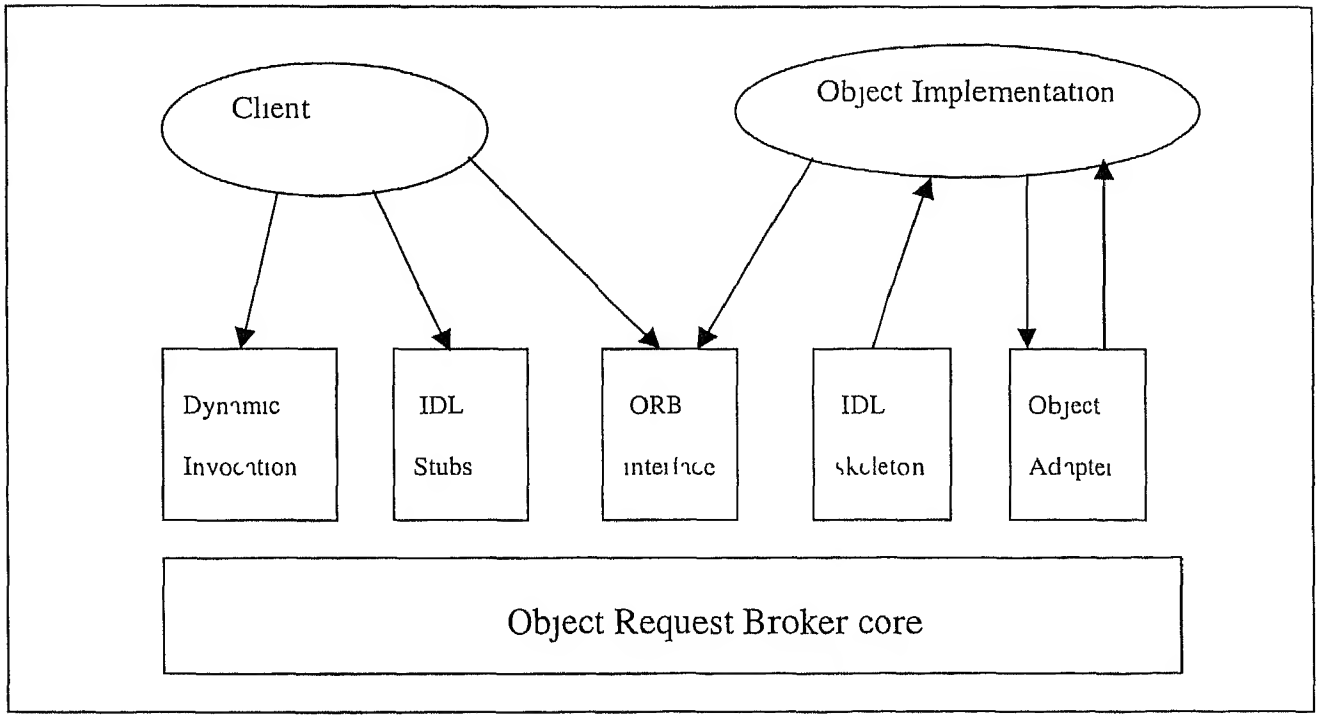


Figure 2 2 CORBA Object Model

2 2 2 Object Model

The CORBA Object model provides objects that isolate the requesters of service from the providers of services An object encapsulates state (attributes) and behavior (operations) and offers access to these only through a well defined interface Objects interact by sending messages to other objects CORBA has atomic types such as long string and boolean and constructed ones such as sequences structures or unions

An interface defines the set of possible operations that can be performed on an object An object satisfies an interface if it can be specified as the target object in each potential request described by the interface An operation of an interface is an identifiable entity with a name optional parameters and a return value that can be performed by the object

The feasibility of using CORBA for remote monitoring of the power distribution automation was studied Though CORBA can serve the purpose it was not used The main reason for not using CORBA was that we wanted the framework to be used to be similar to the framework used in OSI network management Also if one wants to shift to the CORBA framework one can easily do so with the availability of OSI network management framework to CORBA framework converters

2 3 Abstract Syntax Notation 1

Traditional computer communications support human driven remote logon e mail file transfer are being supplemented by new applications requiring complex exchanges of information between computer systems and between appliances with embedded computer chips Some of these exchanges of information continue to be human initiated Others are designed for automatic and autonomous computer to computer communications in support of such diverse activities as

cellular telephones meter reading pollution recording air traffic control control of power distribution and applications in home for control appliances In all these cases there is a requirement for the detailed specifications of the exchanges the computers are to perform and for the implementation of the software to support those exchanges The most basic support for many of these exchanges is provided by the use of TCP/IP and the Internet but other carrier protocols are still in use However the specification of the data formats for messages that are to be passed using TCP requires the design and clear specification of application protocols followed by the implementation of those protocols In a number of industrial sectors ASN 1 is the dominant mode of specifying application protocols [7]

The term ASN 1 can be narrowly used to describe a notation or language called Abstract Syntax Notation One or can be used to describe the notation the associated encoding rules and the software tools that assists its use The features that make ASN 1 important and unique are

- It is an internationally standardized vendor independent platform independent and language independent notation for specifying data structures at a high level of abstraction

- It is supported by rules which determine the precise bit patterns to represent values of these data structures when they have to be transferred over a computer network using encoding rules

- It is supported by tools available for most platforms and several programming languages that map the ASN 1 notation into data structure definition in memory and the defined bit patterns for transfer over a communications line

The Application protocol written in ASN 1 is a binary based protocol specification Protocols can be specified in many ways One fundamental distinction is between character based specification versus binary based specification In the character based specification the protocol is defined as a series

of lines of ASCII encoded text In the binary based specification the protocol is defined as a string of octets or of bits For binary based specification approaches vary from various picture based methods to use of a separately defined notation with associated application independent encoding rules The latter is called the abstract syntax approach This is the approach taken with ASN 1

2 3 1 Application Communication with ASN 1

An application on one machine can talk to an application on another machine by reliably sending octet strings themselves

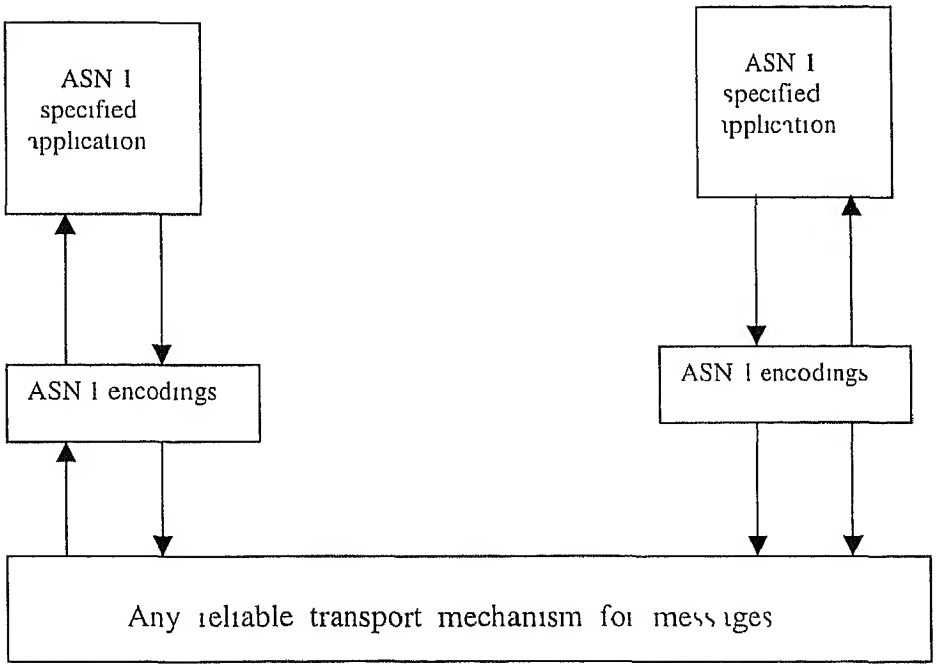


Figure 2 3 Application communication with ASN 1

2 3 2 Object identifiers (OID's)

The object identifier type and its associated hierarchical name space

is heavily used by the protocol specifiers that use ASN 1 It provides a worldwide unambiguous naming scheme Object identifiers are used to identify

a) ASN 1 modules
b) Abstract and transfer syntaxes
c) Managed objects and their attributes
d)

Object identifier tree

The underlying concept for the object identifiers is a tree structure [8] Each object identifier value corresponds to precisely one path from the root down to a leaf with each component of the value notation identifying one of the arcs traversed on this path The Object identifier tree for the thesis is given in Appendix C

2 4 ASN 1 and our Scenario

As seen in the figure 2 3 the reliable transport mechanism for messages can be provided by the TCP features of LabVIEW (refer section 1 3 3 4 of the thesis) As stated earlier object identifiers can be used to identify ASN 1 modules managed objects and their attributes In the case of remote monitoring of power distribution these object identifiers provide a unique way of addressing a particular feeder or a transformer or a switch For example the transformer at a main station can be referred with an object identifier iso 3 6 1 4 1 5305 1 2 1 Similarly a switch at a main station has the object identifier iso 3 6 1 4 1 5305 1 3 2 Similarly the transformer at substation1 can be identified by iso 3 6 1 4 1 5305 2 2 1

The ASN 1 modules that describe the data structures in the case of power distribution monitoring are written for entities like bus switch transformer etc For example an ASN 1 module for a transformer consist of the data structures associated with the transformer Typical elements of this data structure contain the

transformer details the transformer primary and secondary currents and phase angles

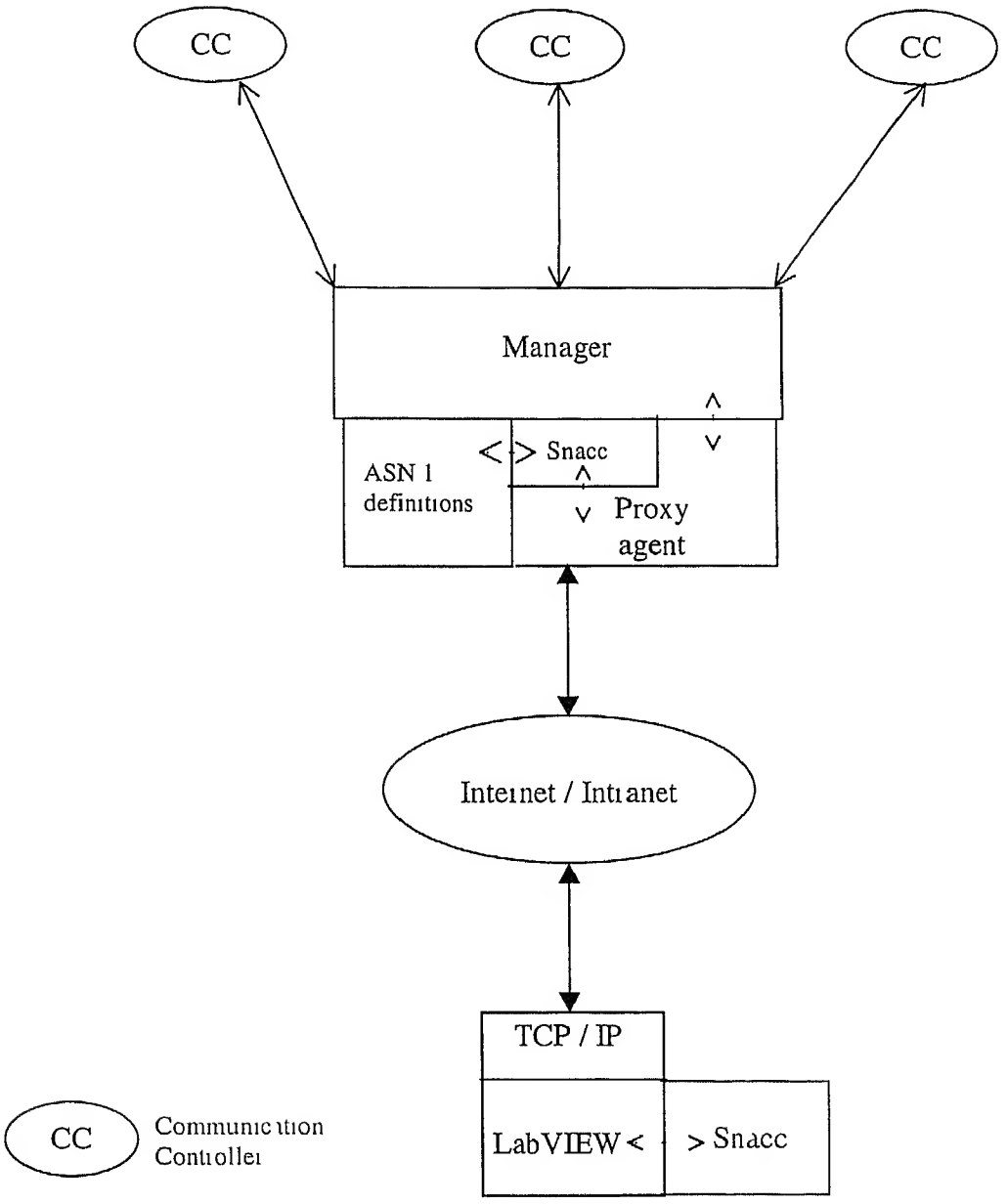


Figure 2 4 Block diagram of the framework proposed in thesis

The ASN 1 produced by the application designer is fed into the compile phase of the ASN 1 tool (Snacc) This maps the ASN 1 into a data structure definition in any

one of a wide range of supported languages including C C++ The application code is written to read and write values from these data structures When an encoding is needed a run time routine is called which uses the information provided by the compile phase about certain aspects of the ASN 1 definition The run time routine encodes the entire message and returns the resulting encodings A similar process is used for decoding The ASN 1 compiler used in the thesis is described in the Appendix A

The OID is passed to the proxy agent which can process the OID The proxy agent in turn gets the required data from the manager The manager has all the information regarding the PDAS which it obtains by polling the communication controllers The proxy agent then encodes the data and is sent back to the GUI The encoded data is decoded using the run time decode routines This can be done using the call library function or the code interface node features of LabVIEW The data is then suitable for analysis or display

2 5 Summary

The ways of achieving open framework in industries that use distributed measurement and control were discussed Description of the approaches like X 700 CORBA ASN 1 were given One of the goals of the thesis is to achieve the open framework in remote monitoring of the power distribution network The approach of using ASN 1 was given in detail with respect to the power distribution automation system

Chapter 3

Virtual Instrumentation for remote monitoring

3 1 Virtual Instrumentation

The idea of virtual instruments [10] is the natural evolution of computer based measurement systems and follows enhancements of the computer hardware as well as the new programming techniques Innovation comprises of the deployment of the computer architecture not only to perform the software programs implementing the measurement procedures but also user oriented functions as well The main goal is in fact to provide a user friendly support to implement and execute the measurement algorithm and the management of the hardware measurement resources

As in conventional computer based measurement and control systems the use of the computer allows for confining the hardware components dedicated to the measurement application to the data acquisition/actuation sub system responsible for sampling the field signals as well as for generating the possible field actuation signals All operations of the measurement procedure are performed in software

The second aspect dealt with by the virtual instruments is the psychological impact on the users which directly affects the distribution of measurement systems Computer based systems even if effective and efficient from the measurement point of view may appear very different from the conventional instruments and measurement workbenches as far as the human user interface is concerned The availability of computational power unused by the real time

measurement procedure in modern computer architecture allows for adopting graphic interfaces to overcome these problems The appearance of real instruments and measurement workbenches is mimicked in the virtual systems so that they resemble the interfaces of their real counterparts

To achieve the above goals the graphic features and modern computers have been exploited Graphic interfaces have been developed based on windowing and iconic interactions to issue commands and observe the operation of the measurement systems as well as the related applications The advanced graphic features of most of the modern personal computers allow for implementing a realistic user interface of the computer based instruments and workbenches very similar to the real ones The name virtual instruments derives in fact from the realistic aspect and operation with respect to the real instruments even if they are internally made in a different way

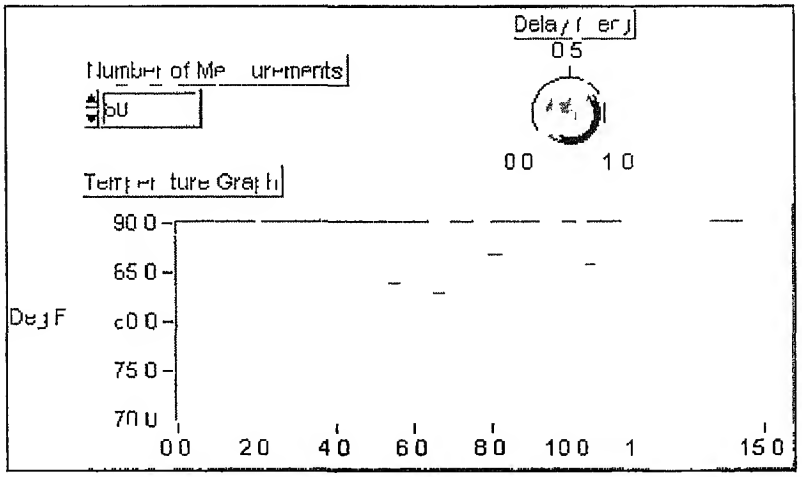


Figure 3 1 The front panel of a dedicated virtual instrument

The front panel presented on the computer screen comprises of the graphic representation of conventional controls (e g buttons selectors) and indicators (e g oscillograms graphs charts lights digital indicators) that are familiar to users in measurement areas Graphic flexibility also allows for easy

merging of the front panels of several instruments with field actuators to create a comprehensive workbench of virtual devices with all the components of the envisioned application This simplifies even more the user interaction with the monitoring and control system since it concentrates all the relevant devices in one single inter related view by discarding the information and the parts of the devices that are not of direct and immediate request The user can operate on the icons as on the corresponding real devices in particular controls are operated by using the pointing device The presentation and user s observation of the measured quantities either on pointer indicators or other visual devices may be less accurate than the numeric view However the visual overview of the pointer and the whole scale helps the user to understand immediately the relative position of the measured quantity in the scale of the values and thus the reasonableness of the observed value This in turn immediately triggers the operator s actions by deploying automatic reactions induced by training in his memory In fact it is often easier to associate actions to patterns in observed images than to read a value and mentally compare it to critical ranges to deduce the opportunity of action

Graphic programming techniques have been imported from computer science to simplify the definition of the measurement procedures by means of object oriented data driven techniques By using these techniques it is possible to describe the measurement procedure without any need for a conventional procedural programming language LabVIEW is one such kind of graphical language

3 2 Graphical Language Programming

In the programming language the sequences of operations on data must be given with a detailed view of the computing architectural typical of scientists and engineers In visual programming data and operations are represented by icons The measurement procedure is obtained by drawing a procedural diagram that defines the data dependencies and operations to be performed Nodes of the

graph are high level operations typical of the numeric processing in measurement as well as operations for visual data presentation and management Icons that are associated to functional blocks visually represent nodes Nodes have input and output connections that allow receiving and delivering operands and results respectively Input nodes connect the graph to the external sources of data while external nodes connect data to display devices or output boards to the field Directed paths represent the data dependencies among operations To simplify the design process huge libraries of functional blocks are available for the widely used development environments for virtual instruments and workbenches including data processing operation In addition to deal with complex measurement procedures hierarchical definition of the graph is usually adopted to structure the definition at different abstraction levels A node may therefore be either a library function or a sub graph

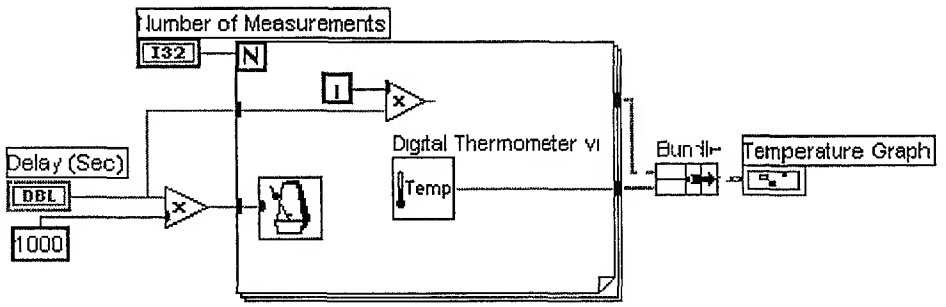


Figure 3 2 The block diagram of the instrument presented in Figure 3 1

A graphic editor allows for creating the measurement procedure graph Editing functions include selection of functional blocks from the library connection functions placement and routing drag and drop cut and paste and much more Customization of the operation parameters is possible according to the characteristics associated with the envisioned icon Data flow is defined by properly connecting the instances of the functional blocks The graphic interface is then associated with the measurement procedure so that inputs and outputs are associated to the corresponding operation blocks

The measurement procedure graph coincides with the conventional block diagram that every measurement designer is accustomed to drawing to describe the overall measurement procedure as well as the details specifying the characteristics of the desired data processing This is independent from the techniques adopted to implement the procedure itself Therefore visual programming allows the measurement problems and issues instead of wasting efforts in computer programming details

The block diagram of the measurement procedure is then translated into a computer executable code by the suitable automatic translation programs available in the virtual system development environment The compiler generates the executable code of the measurement procedure and the function library and stores it in an executable file for subsequent use

3 3 Virtual Instrumentation in Remote monitoring of Power distribution

Conventional object oriented approaches for user interfaces do not mimic the real instruments The real instruments can be mimicked by the use of virtual instrumentation methods The user gets the visual feeling that he is using a real measurement instrument Also it is physically not feasible to carry the measurement equipment upto the place where the measurement has to be taken For example it is difficult to take a cathode ray oscilloscope to the substation to view the waveform If the sufficient number of data points are present the required waveform can be displayed on the virtual instrument

3 4 Important features of LabVIEW in remote monitoring of Power distribution

In the context of remote monitoring of the power distribution network the communication between the user interface and the agent is done using

the TCP/IP suite of protocols The communication features of LabVIEW are utilized As stated earlier LabVIEW has extensive libraries for data analysis and data presentation The rest of the chapter deals with the communication applications used for the thesis

3 4 1 Implementation issues in remote monitoring over Internet using LabVIEW

This section describes the way LabVIEW handles networking and communication features Networking refers to communication between multiple processes The processes can optionally run on separate computers This communication usually occurs over a hardware network such as Ethernet One main use for networking in software applications is to allow one or more applications to use the services of another application For example the application providing services (the server) could be either a data collection application running on a dedicated computer or a database program providing information for other applications

Several protocols are built into LabVIEW some of which are specific to a type of computer LabVIEW uses the following protocols to communicate between computers TCP UDP DDE and OLE Other communication option by LabVIEW includes the System Exec VI which allows executing system level commands

3 5 Client/Server Model

The client/server model is a common model for networked applications In the client/server model one set of processes (clients) request services from another set of processes (servers) For example an application could set up a dedicated computer for acquiring measurements from the real world The computer acts as a server when it provides data to other computers on request It

acts as a client when it requests another application such as a database program to record the data that it acquires

General Model for a Client

The following block diagram shows what a simplified model for a client looks like in LabVIEW

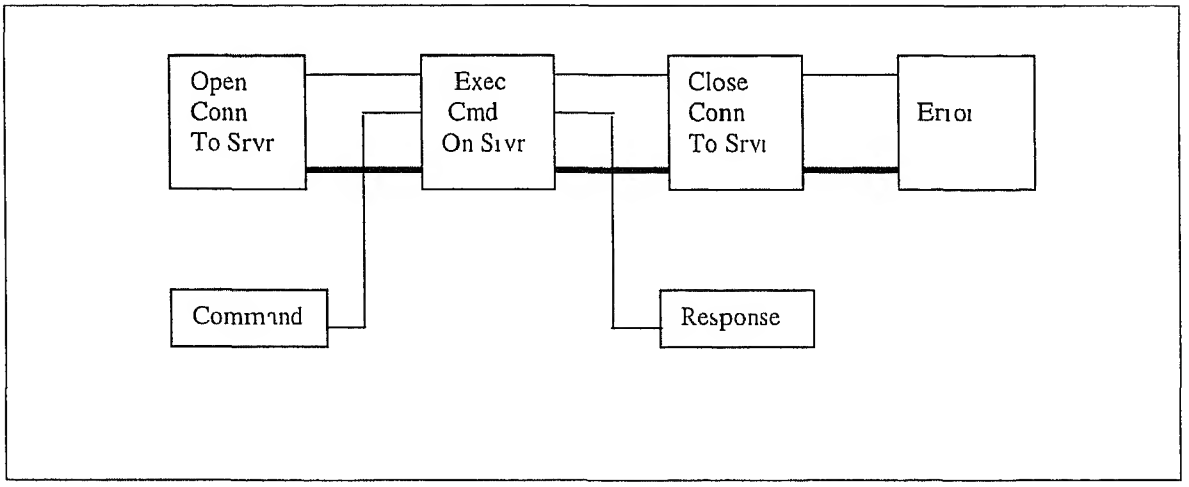


Figure 3 3 General model of a client

In the preceding diagram LabVIEW first opens a connection to a server It then sends a command to the server gets a response back and closes the connection to the server Finally it reports any errors that occured during the communication process For higher performance one can process multiple commands once the connection is open After the commands are executed the connection is closed

General Model for a Server

The following block diagram shows a simplified model for a server in LabVIEW

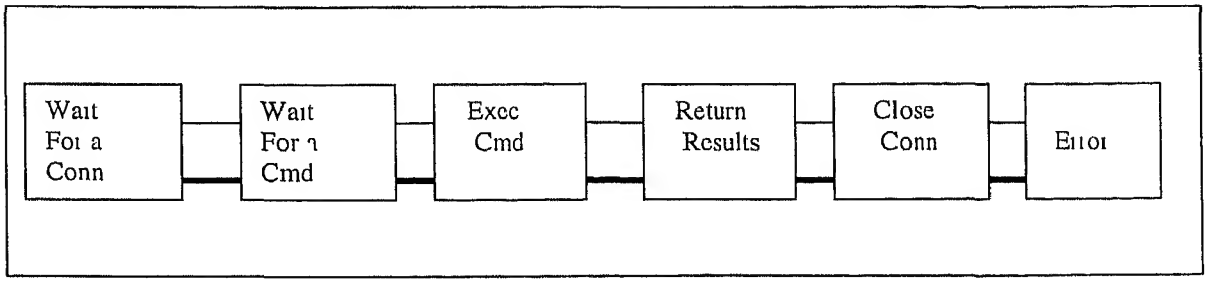


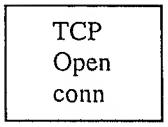
Figure 3 4 General model of a server

In the preceding diagram LabVIEW first initializes the server If the initialization is successful LabVIEW goes into a loop where it waits for a connection Once the connection is made LabVIEW waits to receive a command LabVIEW executes the command and returns the results The connection is then closed LabVIEW repeats this entire process until it is shut down locally by pressing a stop button on the front panel or remotely by sending a command to shut the VI down

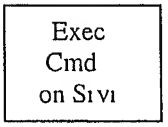
One can increase performance by allowing the connection to stay open so that server can receive multiple commands but this blocks others clients from connecting until the current client disconnects If the protocol supports multiple simultaneous connections one can restructure LabVIEW to handle multiple clients simultaneously

TCP Client Example

The following is a generalized description of how to use the components of the Client block diagram model with the TCP protocol



TCP Open Connection VI is used to open a connection to a server The Internet address of the server as well as the port for the server is to be specified The address identifies a computer on the network The port is an additional number that identifies a communication channel on the computer that the server uses to listen for communication requests



To execute a command on the server the TCP Write VI is used to send the command to the server TCP Read VI can then be used to read back results from the server With the TCP Read VI the number of characters that are to be read is to be given This can be awkward because the length of the response may

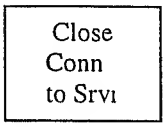
vary The server can have the same problem with the command because the length of a command can vary

The following are several methods you can use to address varying sized commands

1 Precede the command and the result with a fixed size parameter that specifies the size of the command or result In this case read the size parameter and then read the number of characters specified by the size This option is efficient and flexible

2 Make each command and result a fixed size When a command is smaller than the size pad it out to the fixed size

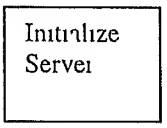
3 Follow each command and result with a specific terminating character To read the data it is to be read in small chunks until the terminating character is found



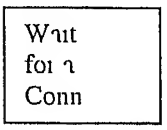
Use the TCP Close Connection VI to close the connection to the server

TCP Server Example

The following discussion explains how TCP can be used to fulfill each component of the general server model



No initialization is necessary with TCP so this step can be left out



The TCP Listen VI is used to wait for a connection The port that is used for communication is to be specified This port must be the same port that the client attempts to connect The TCP Client Example topic provides more information about this VI

Wait for a Cmd

If a connection is established read from that port to retrieve a command As discussed in the TCP Client example the format for commands are to be decided If commands are preceded by a length field the length field is read and then the amount of data indicated by the length field is read

Exec Cmd

Execution of a command should be protocol independent because it is performed on the local computer When finished the results are passed to the next stage where they are transmitted to the client

Return Results

The TCP Write VI is used to return results As discussed in the TCP Client example the data must be in a form that the client can accept

3 6 Communication model in LabVIEW for remote monitoring of the Power distribution automation

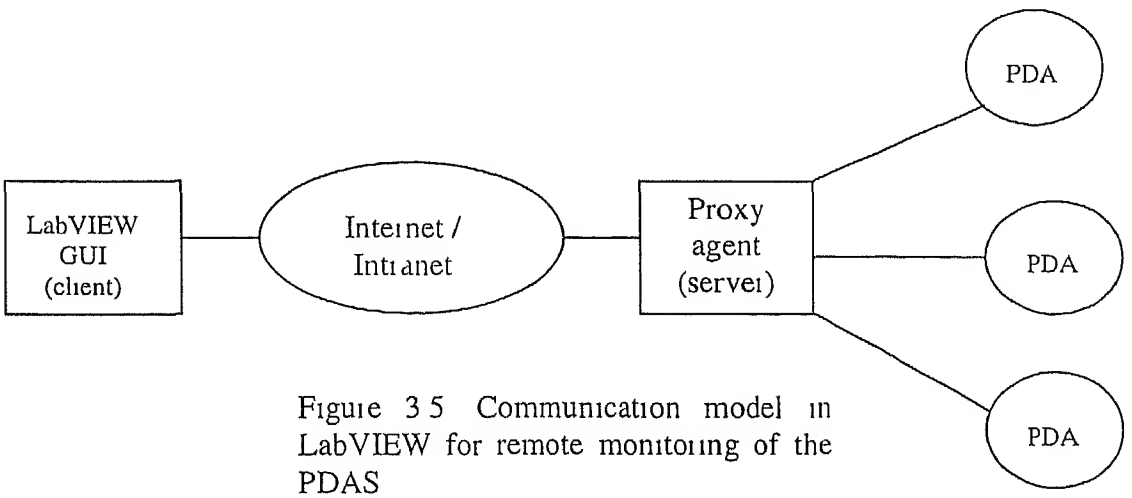


Figure 3 5 Communication model in LabVIEW for remote monitoring of the PDAS

The communication model used for the remote monitoring of the power distribution automation systems is shown in the above figure The GUI forms the client The

model of the client is given in figure 3 3 The proxy agent forms the server The general model of the server is shown in figure 3 4

3 5 Summary

This chapter introduces the evolution of virtual instruments and the its advantages The consequence is the evolution of the G language Use of virtual instrumentation for remote monitoring and diagnostic analysis in the case of power distribution network was given Also the implementation issues of using LabVIEW in remote monitoring over Internet were mentioned The next chapter describes the implementation

Chapter 4

Implementation Details

The aim of the thesis is to configure and tailor a LabVIEW centered instrumentation for monitoring and analysis of distributed real time utility parameter over the TCP/IP based networks The user interface is built for a part of the IITK power distribution automation network The user interface provided the basic elements of the power utility The description of the infrastructure for monitoring the power distribution is shown in figure 1 1

4 1Identification of the substation elements in Power Distribution

A study of the elements of the power utility has been made At the first instance the following elements were of interest

1) Bus
2) Transformer
3) Switch
4) Load

- Bus data consists of a bus number that is assigned to the bus name of the bus the rated voltage of the bus and the voltage of the three phases
- Transformer data consists of the number and the name of the transformer the currents and the phases for the high voltage and low voltage sides and the power rating of the transformer
- Switch data consists of the name and number of the switch the status of the switch and the mode of operation of the switch i e whether the switch can be controlled remotely or locally
- Load data consists of the particular load identification number name of the load the switch number through which the load is connected the load currents and the phase angles

These are the basic elements that are been shown in the user interface Other elements of interest can be added

4 2 Object Identifier Assignment

The concept of the Object Identifiers (OIDs) was described in section 2 3 2 of the thesis Static assignment of the OIDs was used in the thesis The reason behind the static assignment is that in any industrial utility such as the power utility changes in the field do not at a rapid time For example no additional transformer at any particular substation will be added every few months This was the reason for the static assignment of the OIDs

The hierarchical nature of the OID assignment was used in the OID assignment For using the OID name space one has to get registered with the Internet Assigned Numbers Authority IIT Kanpur is registered under the Private Enterprises node The number assigned to IIT Kanpur was PEN 5305 [11] Refer to Appendix C for the Object Identifier tree So all the OIDs start with iso 3 6 1 4 1 5305 The general OID assignment is as follows iso 3 6 1 4 1 5305 a b c

a denotes the substation identifier
- 0 main station
- n substation n

b indentifies the basic entity
- 0 bus
- 1 transformer
- 2 switch
- 3 load

c denotes which b in a

For example the second transformer in main station is denoted by iso 3 6 1 4 1 5305 0 1 2

4 3 ASN 1 modules

After the identification of the elements and the assignment of the Object identifiers the data structures for the elements are specified using ASN 1 syntax These specifications are called ASN 1 modules In the thesis ASN 1 modules were made for the elements identified An example module is shown below and others are given in Appendix B

Bus Module

```
BUS DEFINITIONS  =
   BEGIN
    BusData  =  snacc isPdu  TRUE    [APPLICATION 0] IMPLICIT SET
   {
      busstat  [0] IMPLICIT SEQUENCE OF BusDetails DEFAULT {}
     }
    BusDetails  = [APPLICATION 1] IMPLICIT SEQUENCE
   {
            busno       INTEGER
            busname     IA5String
            vrating     INTEGER
            va          REAL
            vb          REAL
            vc          REAL
            time        UTCTime
      }

  END
```

An example ASN 1 module

The above ASN 1 module corresponds to a Bus module BusDetails is the sequence of various bus attributes The module starts with the BEGIN command and finishes by an END command For more details of the syntax of writing the modules refer to [7]

4 4 ASN 1 encoding and decoding routines

The ASN 1 encoding and decoding routines described in this section are for the ASN 1 type BusData which is in the Bus module given in the previous page The header file generated by the compiler is given in Appendix D This file contains the routines for all the ASN 1 types used in the particular module

Encode Routine

BencBusData ( )

This routine takes two arguments the buffer to encode the value into and a pointer to the instance of the ASN 1 type that is to be encoded

Decode Routine

BdecBusData ( )

This function takes four arguments the buffer to hold the BER value to be decoded a pointer to space allocated for the type being decoded a parameter maintaining the running total of the number of octets that have been decoded and a environment parameter for error management

Print Routine

PrintBusData ( )

All the print routines take similar parameters The print routine writes the given value to the file The printed value is indented by indent spaces The values are printed in ASN 1 value notation style

Free Routine

FreeBusData ( )

The free routines will free all the components named type For exmaple the FreeBusData routine will free all the components of the given BusData value but not the BusData value itself

4 5 Implementation

The setup for the implementation is given in figure 2 4 We can observe from the figure that the ASN 1 compiler has to be present at both the ends i e at the user interface and the agent as well The ASN 1 compiler (Snacc) which was used in the thesis was installed in linux but was not possible in windows in which the LabVIEW operates The agent was a LabVIEW server which is connected to the user interface over the network On request the agent sends the ASN 1 decoded file to the user interface More details of the implementation were given in the following two sections

4 6 Front Panel

Development of the user interface is done in a manner that if an user selects an option at the layout diagram another panel pops up on the screen This feature is achieved by making the selected VI as a sub VI and using the VI setup options [2]

The first front panel (figure 4 1) of the user interface is a layout containing the main substation and various substations The user can select any of the substations which leads into that substation The panels for substations 1 and 2 are shown in figure 4 2 4 3 The substation layout is presented there with the necessary labels As the user can see all the power devices such as the bus bars switches transformers etc on the layout diagram he can click on the particular label The front panel of a bus (figure 4 4) has a label containing the bus number

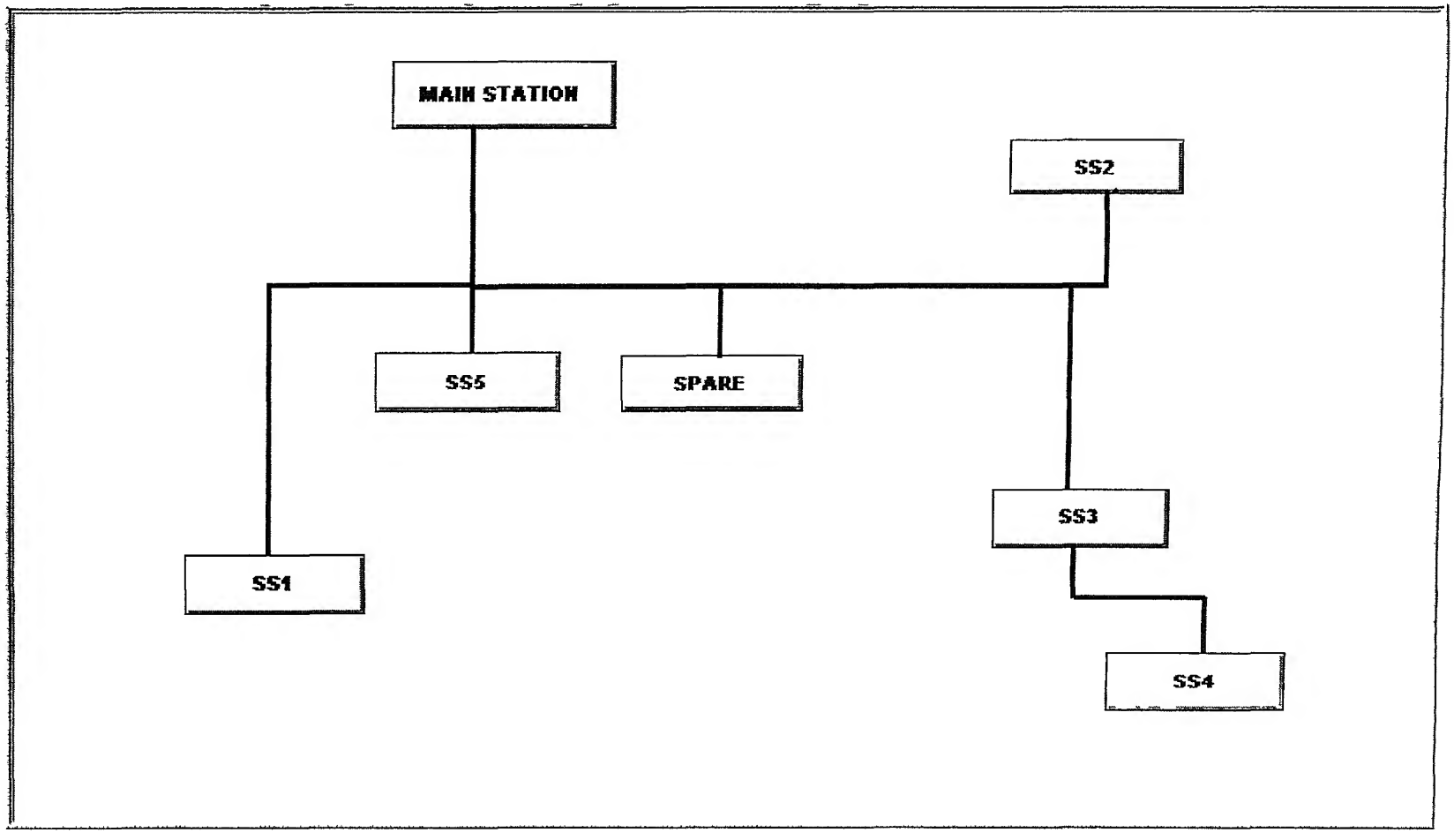


Figure 4 1 Main Front panel

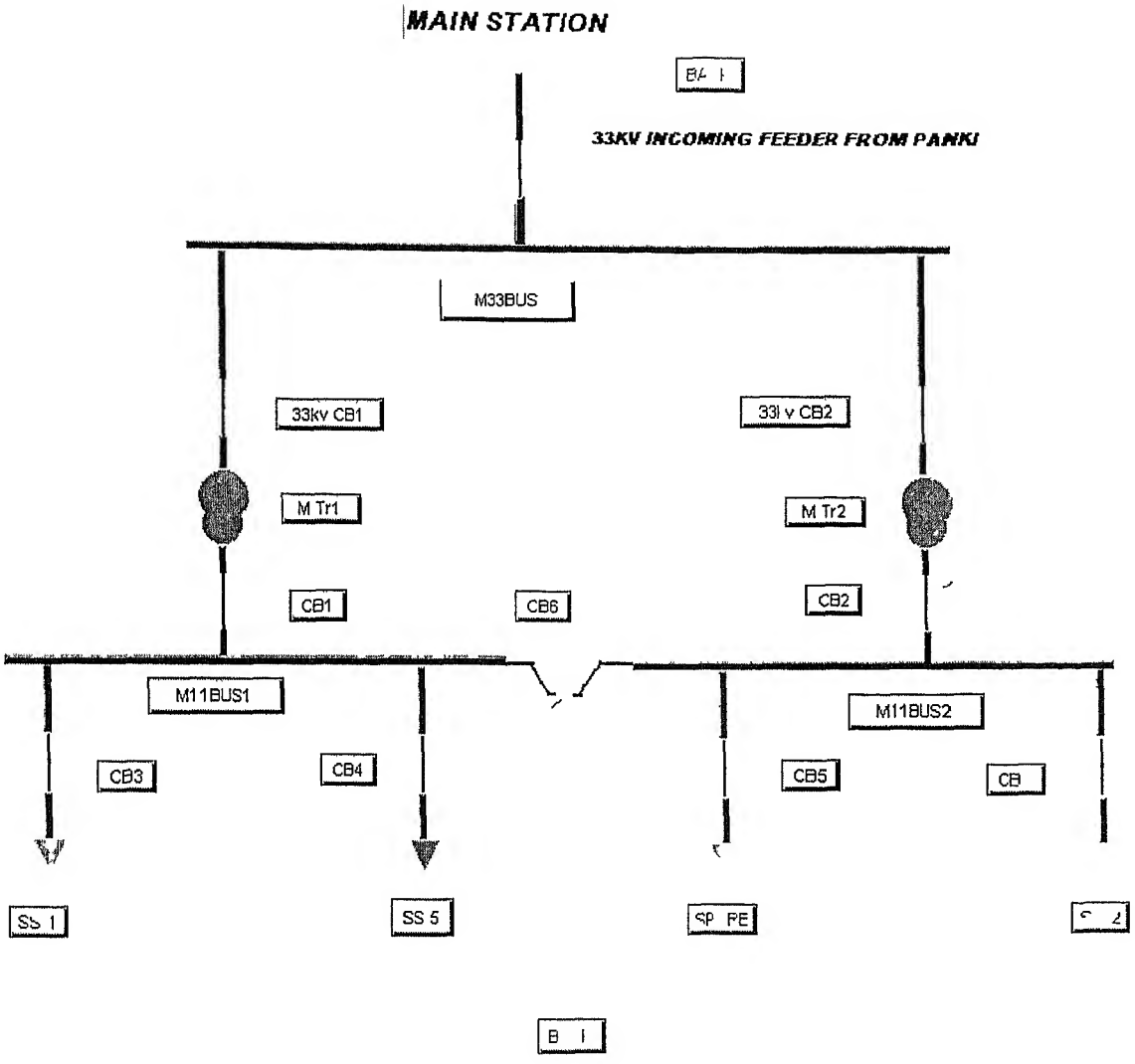


Figure 4 2 Front panel of the main substation

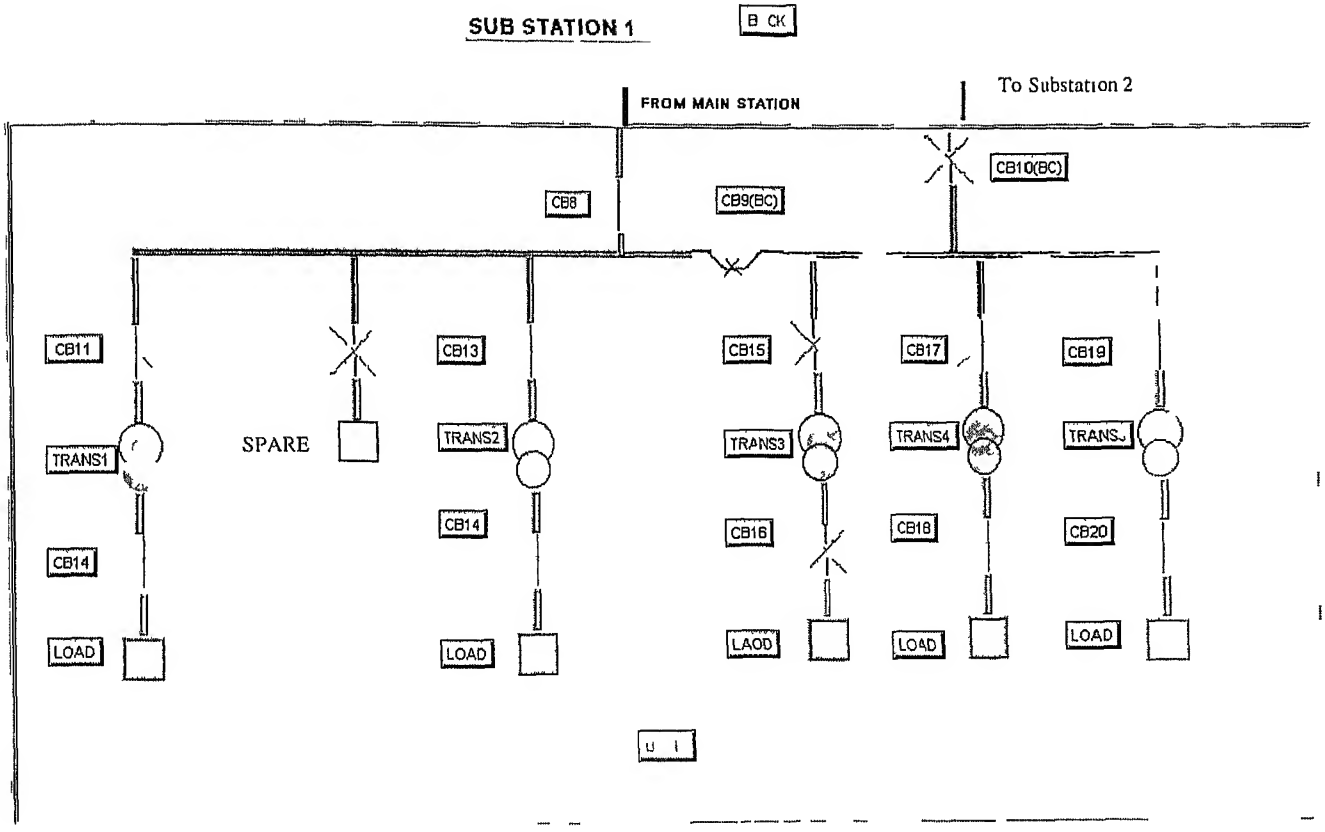


Figure 4 3 Front panel of the substation 1

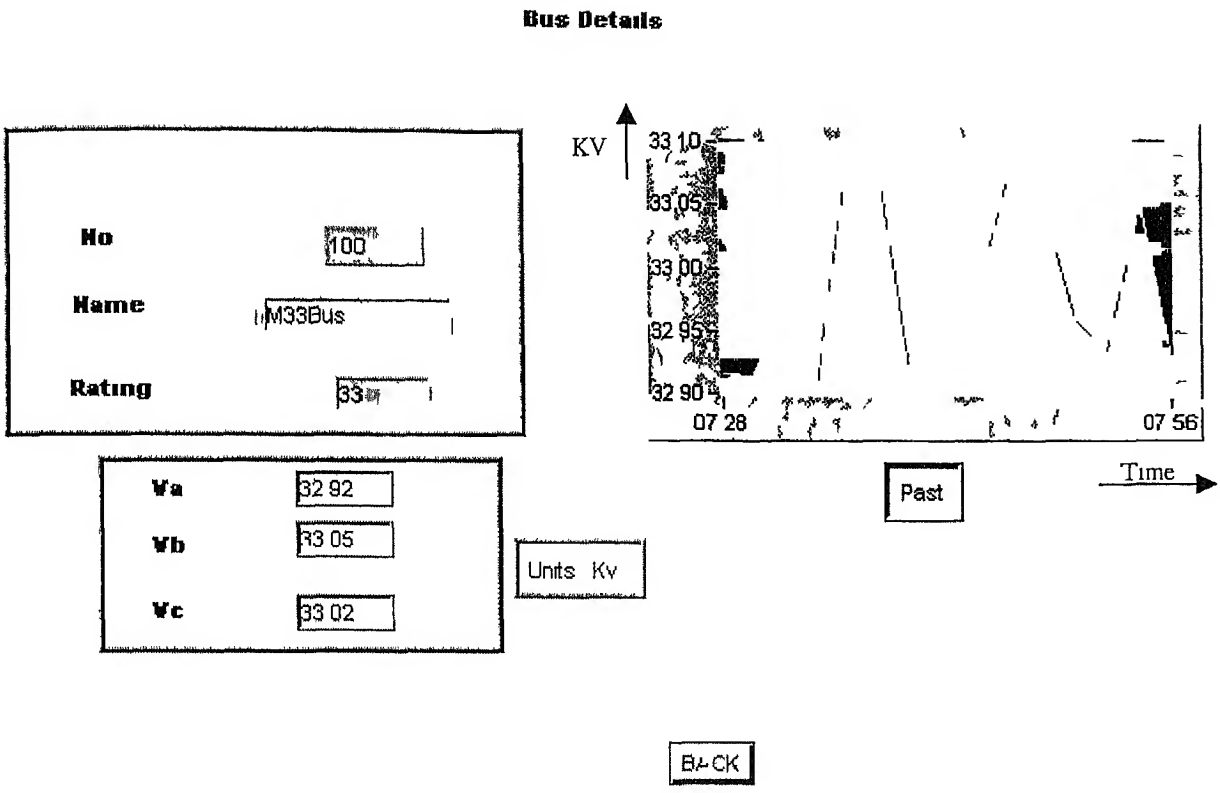


Figure 4 4 Front Panel for Bus Details

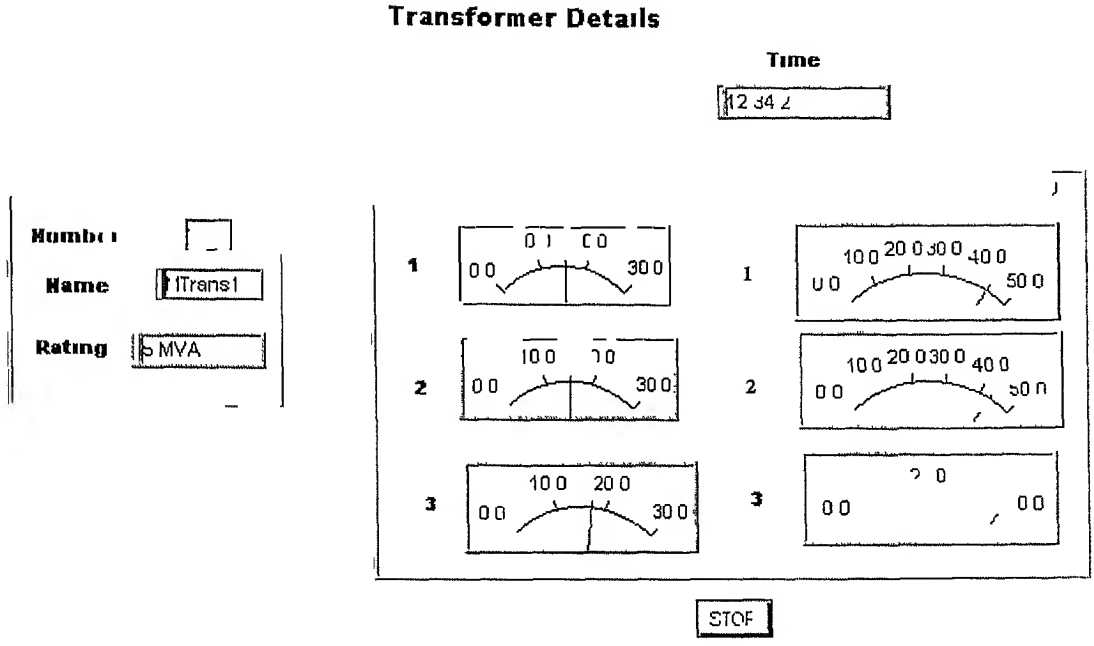


Figure 4.5 FrontPanel for transformer details

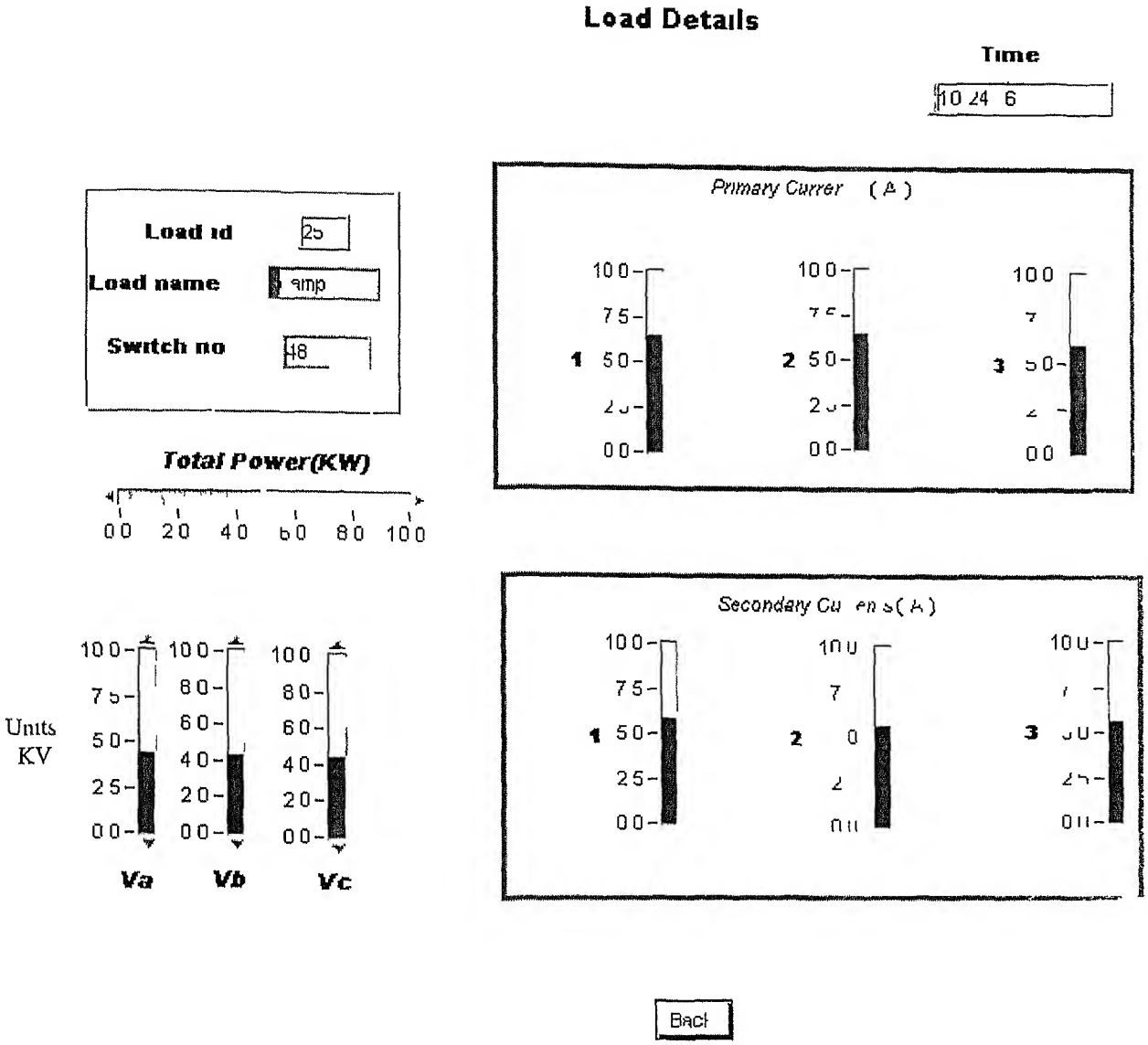


Figure 4 6 FrontPanel for load details

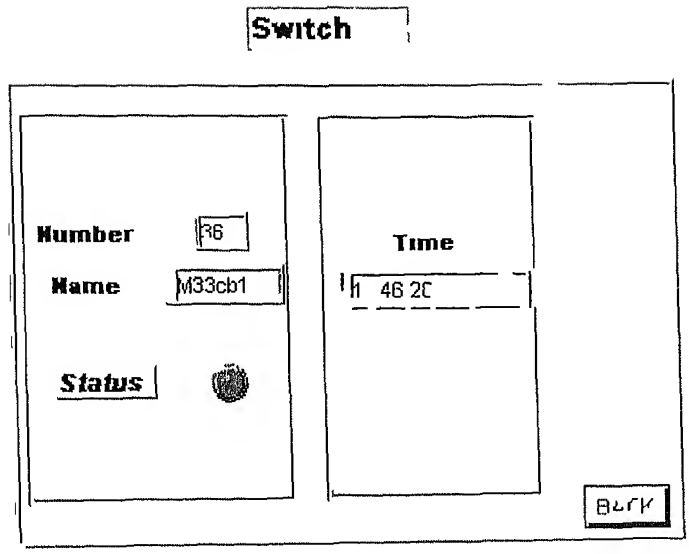


Figure 4 7 Front Panel for Switch details

bus name and the voltage rating of the bus It also has digital indicators for display of phase voltages The front panel of the bus has also a graph which gives a past history of the bus voltage The front panel of a switch (figure 4 7) contains switch name switch number and the status of the switch is indicated by a LED The front panel of a transformer (figure 4 5) has the transformer details viz transformer name transformer number and its power rating Analog meters indicating the primary and secondary currents as well as the phase angles

One has to find the button which the user had selected For this all the controls are built as a cluster The use of clusters reduces wire clutter Cluster will have elements belonging to either controls or of indicators only All the elements inside the cluster are ordered numerically This feature enables to access the individual elements in the block diagram It is assumed that the user will press a single control on the front panel The cluster of identically typed elements is converted into a 1D array of elements of the same type using the cluster to array function As the cluster consists of boolean controls the output of the function will be an array of boolean controls Using the Search 1D array function index corresponding to the control selected by the user is found

Defined Controls

LabVILW provides numeric digital boolean controls and indicators LabVIEW user can choose the type of the control or indicator needed from the controls palette of the front panel Some of the controls and indicators are digital control and indicator knob dial meter gauge horizontal and vertical slides Light Emitting diodes etc

In the power utility various switches such as circuit breakers bus couplers isolators etc are represented by an unique symbol LabVIEW provides the facility for building user defined controls or indicators Using the custom control features various controls and indicators have been built

4 7 Block Diagram

Behind the front panel in the LabVIEW program there exists a block diagram The block diagram is the source code for this programming language The user selects on a particular selector on the front panel to view the necessary data The corresponding object identifier is used to identify the data that is requested A client program is started The client program is built using the TCP library An example of how to build a client program in LabVIEW is given in Chapter 3 As described in section 3 5 of the thesis there is a problem of varying length commands This problem is solved by terminating the object identifier with a known character / The server looks for this terminating character and then accordingly solves the problem

In the TCP Open connection the IP address and the port of the remote host is specified If the connection is established with the remote host a connection ID is returned The connection ID is a network connection refnum that uniquely identifies the TCP connection This connection ID is used to refer to this TCP connection for subsequent VI calls The TCP Write VI is used to write the string data in to the specified TCP connection The string data is the object identifier TCP Read VI received up to the bytes to read from the TCP connection returning the results

The client VI gets the encoded data from the TCP Read VI The received data ASN 1 is BER encoded and so has to be decoded After this pass the corresponding decode routines are to be called The decode routines write the decoded data into a file This file is read again and the data is separated out accordingly to be given as inputs for the various indicators on the front panel

4 8 Summary

In this chapter the implementation details of the graphical user interface are described Assignment of the object identifiers and the data structures

exchanged were given Development of the front panel and appearance were described and the various front panels were shown The logical sequence of events were presented in the block diagram The next chapter concludes the work and discusses the future scope of the work

Chapter 5

Summary and Future work

Geographically distributed utilities such as power distribution gas water etc have not seen the widespread use of networked installation for monitoring and control A monitoring and control network for the power distribution automation was developed at IIT Kanpur There was a need for developing a user interface which has live instrument display for remote monitoring of the power distribution utility Also there was a need for open Internet compatible GUI

The following were accomplished in the thesis

- We have explored LabVIEW for remote diagnostic analysis of installations in the power distribution automation network
- Identification of the substation elements in the power distribution network that are being shown in the GUI
- Implementation of the communication with remote systems on the Internet using LabVIEW s TCP/IP features
- Designing an open framework approach for the representation of the data structures that are to be exchanged between the LabVIEW based GUI and the agent ASN 1 was chosen for the open framework
- Present implementation of Smee tool was done in Linux based remote system Coding to produce the encodings and decodings were written
- A simple implementation was done using simulated data for the development of

the GUI for remote visualization of the electric power substation parameters

The source files for the GUI are stored in c:\gui\ in the PC as well as submitted to the ERNET Lab simulator vi is the main vi for the GUI and is to be run in Run Continously mode One has to specify the IP address and the TCP port of the server in client vi To run the demo one has to run the server vi also

Scope for further work

The user interface was developed for the remote monitoring of the power distribution utilities A more sophisticated interface can be developed to address particular objects of interest With reference to figure 2 4 the proxy agent has to be configured Work in this direction is done as a separate thesis [13] There is also a need for developing a special RTU which can act as a remote troubleshooter and which can be used for certification of new installation etc

References

[1] Apurva Gurwak Development of a Field bus compatible Remote data acquisition and controller system M Tech Thesis IIT Kanpur 1994

[2] National Instruments LAbVIEW Tutorial 1996

[3] Kang B Lee Richard D Schneeman Internet Based Distributed Measurement and control Applications IEEE Instrumentation & Measurement Magazine June 1999

[4] National Instruments Communications VI Reference Manual 1996

[5] Bela Ban Towards an Object Oriented Framework for Multi domain management IBM Zurich Research Laboratory Dec 1995 http //www cs cornell edu/Info/People/bba/papers html

[6] Sean Baker Vinny Cahill and Paddy Nixon Bridging boundaries CORBA in perspective IEEE Internet Computing September October 1997

[7] John Larmouth *ASN 1 Complete* Open Systems Solutions May 1999

[8] Marshall T Rose *The Simple Book* Prentice Hall 1995

[9] Alwyn Langsford Jonathan D Moffett *Distributed Systems Management* Addison Welsley 1992

[10] Lorenzo Cristaldi Alessandro Ferrero and Vincenzo Piuri Programmable Instruments Virtual Instruments and Distributed Measurement Systems IEEE Instrumentation and Measurement Magazine September 1999

[11] ftp //ftp isi edu/in notes/iana/assignments/enterprise numbers Network Management Private Enterprise Codes

[12] Michael Sample *Snacc 1 3 A High Performance ASN 1 to C/C++/IDL Compiler* Department of Computer Science University of British Columbia Canada

[13] Harish Bishnoi An Object Request Brokering approach for the monitoring of multiple Power Distribution Automation System over Internet M Tech Thesis February 2000

Appendix A

Sample Neufeld ASN 1 to C/C++ Compiler (Snacc)

Snacc [12] compiles ASN 1 modules into C C++ or type tables The generated C or C++ code contains equivalent data structures and routines to convert values between the internal(C or C++) representation and the corresponding BER (Basic Encoding Rules) format The ASN 1 data structure language can specify complex types such as lists and recursively defined types BER data values are defined independently of any computer architecture providing a universal data value representation that is useful for sharing data in heterogeneous networks

The process of converting an ASN 1 value from its C or C++ representation into an equivalent BER data value is called encoding and the reverse process is called decoding

Some of the snacc s features include

- parses CCITT ASN 1 90 including subtype notation
- can compile and link interdependent ASN 1 modules(IMPORTS AND EXPORTS)
- some X 400 and SNMP macros are parsed
- value notation is parsed OBJECT IDENTIFIERS INTEGERS AND BOOLEANS are translates to C/C++ values
- ANY DEFINED BY types are supported via the SNMP OBJECTIVE TYPE macro

The main passes of the compiler are executed in the following order

1 parse useful types ASN 1 module
2 parse all user specified ASN 1 modules
3 link local and imported type references in all modules
4 parse values in all modules
5 link local and imported value references in all modules

6 process any macro types

7 normalize types

8 mark recursive types and signal any recursion related errors

9 check for semantic errors in all modules

10 generate C/C++ type information for each ASN.1 type

11 sort the types from least dependent to most dependent

12 generate C C++ IDL or type table

Code Generation

If the target language is C for each ASN.1 type an equivalent C data type a BER encoding routine a printing routine and a freeing routine will be generated These are in the .h and .c files

The basic outline diagram (figure A.1) of Snacc functionality is given in the next page

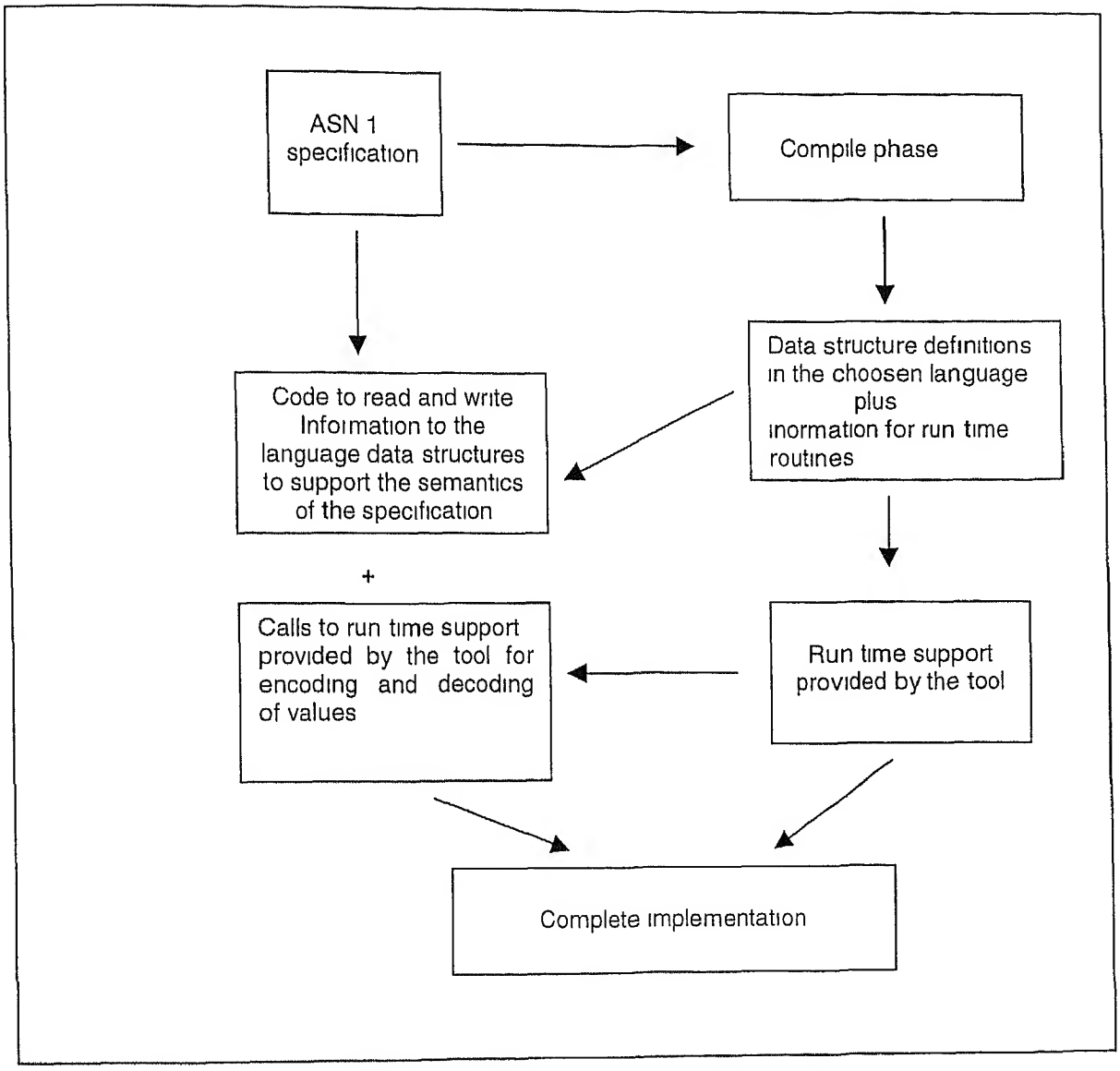


Figure A 1 Basic Functionality of Snacc

Appendix B

ASN 1 modules

Transformer Module

```
TRANS DEFINITIONS ::=
        BEGIN
    TransData ::= snacc isPdu TRUE    [APPLICATION 0] IMPLICIT SET
        {
            transstat [0] IMPLICIT SEQUENCE OF TransDetails DEFAULT { }
        }
     TransDetails ::= [APPLICATION 1] IMPLICIT SEQUENCE
      {       tino        INTEGER
              tname       IA5String
              hvbusno     INTEGER
              lvbusno     INTEGER
              ia1         REAL
              ib1         REAL
              ic1         REAL
              ia2         REAL
              ib2         REAL
              ic2         REAL
              pa1         REAL
              pb1         REAL
              pc1         REAL
              pa2         REAL
              pb2         REAL
              pc2         REAL
              time        UTCTime
        } END
```

Switch Module

```
SWITCH DEFINITIONS  =

   BLGIN


SwitchDetail  =  switchesPdu  TRUE     [APPLICATION 0] IMPLICIT SET
     {
       switchstat   [0] IMPLICIT SEQUENCE OF SwitchDetails DEFAULT { }
    }
SwitchDetails  = [APPLICATION 1] IMPLICIT SEQUENCE
{
              swno          INTEGER
           swname          IA5String
              status         INTEGER
   1 denotes ON  0 denotes OFF
           time           UTCTime
    }

   LND
```

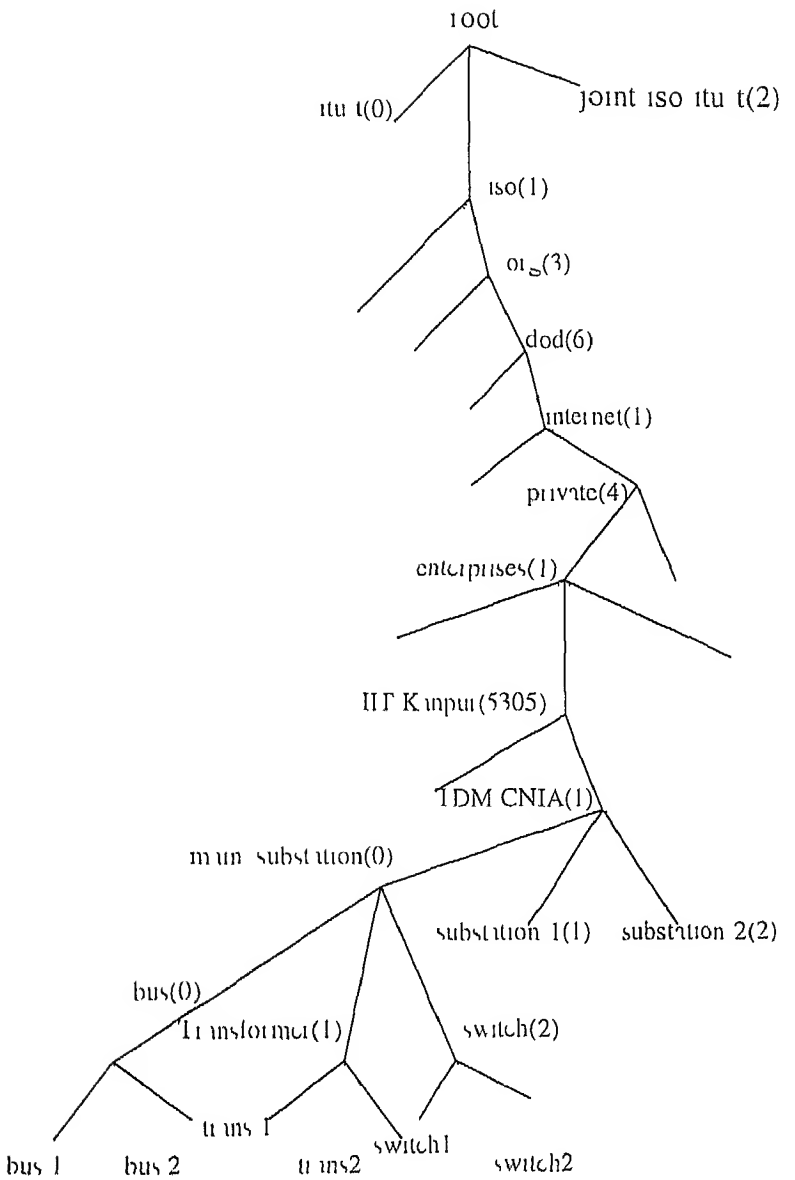
Load module

```
LOAD DEFINITIONS ::=

  BEGIN

   LoadData ::= sequence isPdu TRUE    [APPLICATION 0] IMPLICIT SET
     {
      loadstat  [0] IMPLICIT SEQUENCE OF LoadDetails DEFAULT {}
     }

   LoadDetails ::= [APPLICATION 1] IMPLICIT SEQUENCE
     {
          busno       INTEGER
          busname     IA5String
          loadid      INTEGER
          loadname    IA5String
          swno        INTEGER
          ia1         REAL
          ib1         REAL
          ic1         REAL
          ia2         REAL
          ib2         REAL
          ic2         REAL
          va          REAL
          vb          REAL
          vc          REAL
          pa1         REAL
          pb1         REAL
          pc1         REAL
          pa2         REAL
          pb2         REAL
          pc2         REAL
          time        UTCTime
  }

   END
```

CENTRAL LIBRARY
I I T, KANPUR
Acc No A 130799

Appendix C

Object identifier tree



For example the Object identifier for bus1 in the main station is formed by traversing the tree and is iso 3 6 1 4 1 5305 1 0 0 1

Appendix D

Snacc compiler generated files

If the ASN.1 module is applied to the snacc compiler and if the target language of the compiler is C then the compiler generates a .h and a .c file for the corresponding ASN.1 module. This appendix gives the .h and .c files generated by the snacc compiler for the Bus module. This module is given in chapter 4. The .h file contains the function prototypes.

Generated .h file

```
/*
 *  bus.h
 *
 *  BUS - ASN.1 module C type definitions and prototypes
 *
 *  This .h file was generated by snacc on Thu May 02 22:10:53 2000
 *
 *  UBC snacc written compiler by Mike Sample
 *
 *  NOTE: This is a machine generated file - editing not recommended
 */

#ifndef _bus_h_
#define _bus_h_

typedef struct BusDetails /* [APPLICATION 1] IMPLICIT SEQUENCE */
{
   AsnInt busno /* INTEGER */
   AsnOcts busname /* OCTET STRING */
   AsnInt voltang /* INTEGER */
   AsnReal va /* REAL */
   AsnReal vb /* REAL */
   AsnReal vc /* REAL */
   AsnOcts time /* OCTET STRING */
} BusDetails
```

```
AsnLen BEncBusDetailsContent PROTO ((BUF_TYPE b, BusDetails *v));

void BDecBusDetailsContent PROTO ((BUF_TYPE b, AsnTag tagId0, AsnLen
elmtLen0, BusDetails *v, AsnLen *bytesDecoded, ENV_TYPE env));


void PrintBusDetails PROTO ((FILE *f, BusDetails *v, unsigned short int indent));
void FreeBusDetails PROTO ((BusDetails *v));


typedef AsnList BusDataSeqOf; /* SEQUENCE OF BusDetails */

AsnLen BEncBusDataSeqOfContent PROTO ((BUF_TYPE b, BusDataSeqOf *v));

void BDecBusDataSeqOfContent PROTO ((BUF_TYPE b, AsnTag tagId0, AsnLen
elmtLen0, BusDataSeqOf *v, AsnLen *bytesDecoded, ENV_TYPE env));


void PrintBusDataSeqOf PROTO ((FILE *f, BusDataSeqOf *v, unsigned short int
indent));
void FreeBusDataSeqOf PROTO ((BusDataSeqOf *v));


typedef struct BusData /* [APPLICATION 0] IMPLICIT SET */
{
  BusDataSeqOf *busstat; /* [0] IMPLICIT BusDataSeqOf DEFAULT */
} BusData;

AsnLen BEncBusData PROTO ((BUF_TYPE b, BusData *v));

void BDecBusData PROTO ((BUF_TYPE b, BusData *result, AsnLen
*bytesDecoded, ENV_TYPE env));
AsnLen BEncBusDataContent PROTO ((BUF_TYPE b, BusData *v));

void BDecBusDataContent PROTO ((BUF_TYPE b, AsnTag tagId0, AsnLen
elmtLen0, BusData *v, AsnLen *bytesDecoded, ENV_TYPE env));

void PrintBusData PROTO ((FILE *f, BusData *v, unsigned short int indent));
void FreeBusData PROTO ((BusData *v));

#endif /* conditional include of bus.h */
```

A 130799

A 130799

Date Slip

This book is to be returned on the date last stamped